MEDAPs: secure multi-entities delegated authentication protocols for mobile cloud computing

著者	ZHANG Lei, WEI Lifei, HUANG Dongmei, ZHANG Kai, DONG Mianxiong, OTA Kaoru
journal or publication title	Security and Communication Networks
volume	9
number	16
page range	3777-3789
year	2016-05-12
URL	http://hdl.handle.net/10258/00009448

doi: info:doi/10.1002/sec.1490

RESEARCH ARTICLE

MEDAPs: Secure Multi-Entities Delegated Authentication Protocols for Mobile Cloud Computing†

Lei Zhang[12], Lifei Wei[1*], Dongmei Huang[1], Kai Zhang[3], Mianxiong Dong[4], Kaoru Ota[4]

1. College of Information Technology, Shanghai Ocean University, Shanghai, China
2. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
3. Department of Computer Science and Software Engineering, East China Normal University, Shanghai, China
4. Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran, Japan

ABSTRACT

Since the technology of mobile cloud computing has brought a lot of benefits to information world, many applications in mobile devices based on cloud have emerged and boomed in the last years. According to the storage limitation, data owners would like to upload and further share the data through the cloud. Due to the safety requirements, mobile data owners are requested to provide credentials such as authentication tags along with the data. However, it is impossible to require mobile data owners to provide every authenticated computational results. The solution that signers' privilege is outsourced to the cloud would be a promising way. To solve this problem, we propose three secure multi-entities delegated authentication protocols (MEDAPs) in mobile cloud computing, which enables the multiple mobile data owners to authorize a group designated cloud servers with the signing rights. The security of MEDAPs is constructed on three cryptographic primitive identity-based multi-proxy signature (IBMPS), identity-based proxy multi-signature (IBPMS), and identity-based multi-proxy multi-signature (IBMPMS), relied on the cubic residues, equaling to the integer factorization assumption. We also give the formal security proof under adaptively chosen message attacks and chosen identity/warrant attacks. Furthermore, compared with the pairing based protocol, MEDAPs are quite efficient and the communication overhead is nearly not a linear growth with the number of cloud servers.

KEYWORDS

Mobile Data Owners, Delegated Authentication, Multi-Authentication, Provable Secure, Mobile Cloud Computing

***Correspondence**

*Corresponding author. Email: Lfwei@shou.edu.cn

Received . . .

†This paper is supported by National Natural Science Foundation of China (Grant No. 61402282 and 61272098), 973 Program (Grant no. 2012CB316206), Shanghai Youth Talent Development Program (Grant No. 14YF1410400), Youth Scholars of Shanghai Education (Grant No. ZZHY14025), Open Foundation of State key Laboratory of Networking and Switching Technology of BUPT (Grant No. SKLNST-2013-1-12), ECNU Foundation for Graduate Student Scientific Innovation (Grant No. YJSKC2015-30), Open Foundation of Key Laboratory of Digital Ocean Science and Technology (KLDO201403), and JSPS KAKENHI (No. 26730056, 15K15976), JSPS A3 Foresight Program.

1. INTRODUCTION

With the emerging of cloud computing, cloud-based applications have been booming in recent years[1]. Many applications based on the outsourced computation have attracted the researchers and engineers from academy and industry on account of the cloud's powerful storage and computational capability. Most of the cloud users are using the mobile devices connecting the cloud through wireless (sometimes wired) networks [2, 3]. In contrast with the traditional computation, the cloud users have lost the physical control on their data in cloud computing [4, 5].

As it known, cloud servers are controlled by the cloud service providers in logically but distributed in physics all over the world through the networks [6]. Sometimes, the servers in the cloud might be compromised by hackers to provider unauthentic data and computational results to cloud users. Thus, these kinds of outsourced computation performed on the untrusted cloud servers have restricted the mobile cloud computing to some extent. In some scenarios, the mobile data owners upload their records to the cloud for further sharing among data consumers. Taking the mobile sensors for example, the sensors in the oceans report the temperature, salinity, and pH values in the monitor area and upload to the cloud for further analysis and data sharing. In order to keep the data authentication and integrity, the data owners are encouraged to upload credentials such as authentication tags together with the corresponding data as an evidence to show the validation of the data in the cloud servers. To achieve this target, some related work has tried to build such authentication tags [7, 8]. Moreover, the data consumers are expected to fetch the authenticated data, but also would like to compute and aggregate on these authenticated data [9] relying on the cloud servers' capability.

However, in our cases, the mobile data owners can not achieve onlined connecting to the cloud [10] since they can not afford such an excessive battery consuming in data transmission. In addition, the mobile data owners can not afford too much data computational burden from different computation requests in demand.

There are straightforward solutions that seem to come over the above obstacles. One of the solutions is if the mobile data consumers is allowed to fetch the original data only, the mobile data owners store the whole the authenticated tags in advance. However, it is impossible for the mobile data owners due to the various computation requests from cloud consumers. In most cases, the authenticated data need to be calculated or transformed by the various requests. It is another solution that the computation requests need to be forwarded to the responsible mobile data owners, which results in considerable computational overhead and communication overhead due to the offline of the mobile data owners. In order to reduce the computational cost and communication delay, it is a simple improvement that the mobile data owners directly distribute the secret keys to cloud servers. Thus, the authentication service has been achieved on behalf of the mobile data owners since the cloud servers independently finish the data authentication. However, it also brings severe security risks that the remote cloud servers might have been broken down by the adversaries, which leads to the keys leaked out.

Digital signature are always providing the properties of authenticity and non-repudiation in the communications [11]. Ordinary signature may be unavailable due to the existential unforgery property. Homomorphic signature [12] might be a feasible solution which achieves either additional operations or multiply operations. A number of operations such as data comparison, non-linear calculation and SQL query [13] are beyond the homomorphic signature though it often brings considerable computational cost [12]. As a result, the technique that allows the data owners conditionally outsource their authenticated privilege to the cloud would be a promising way [14].

To address the above problems, we build three secure multi-entities delegated authentication protocols (**MEDAPs**) for mobile cloud computing. According to the novel IBMPS scheme and IBMPMS scheme, the mobile data owners could authorize the authenticated rights to several specified cloud servers which would achieve the authenticated tags' generation in a collaborating way on behalf of the mobile data owners. In addition, our **MEDAPs** allow the mobile data owners to conditionally authorize to the cloud servers. Moreover, the **MEDAPs** are quite efficient compared with other bilinear pairing based protocols.

Our work is a significant effort towards delegated authentication considering multiple mobile data owners and multiple cloud servers to offer authentication service in a mobile cloud model. The contributions can be mainly summarized in following four aspects.

Security Comm. Networks 2015; **00**:1–18 © 2015 John Wiley & Sons, Ltd.
DOI: 10.1002/sec
Prepared using secauth.cls

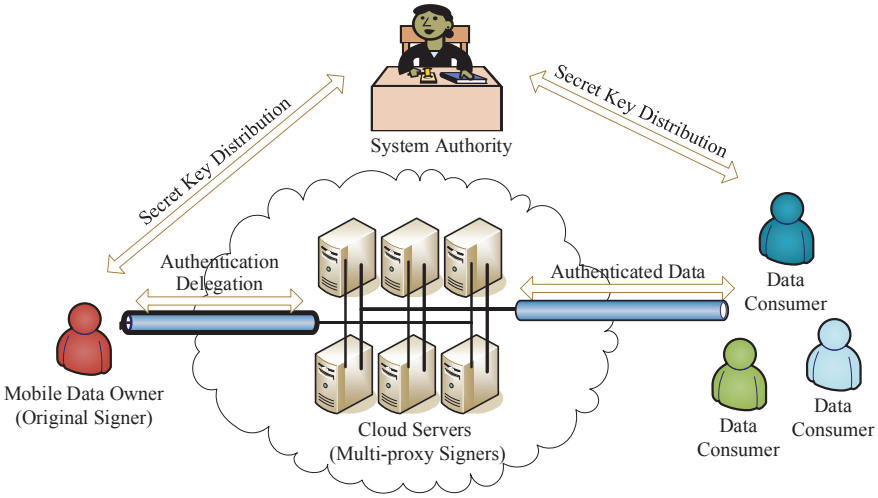

Figure 1. A multi-cloud servers delegated authentication in mobile cloud computing

- Firstly, we propose an IBMPS scheme in cryptography based on cubic residues that can be proved secure under the integer factorization assumption in the random oracle model.
- Secondly, we propose an efficient and multi-entities secure delegated authentication protocol (**MEDAP-I**), as a significant application of our IBMPS scheme, to offer authenticated data and computational results to data consumers by the designated cloud servers on behalf of the data owners.
- Thirdly, considering various situations and preventing the single point of failure, we propose two multi-entities delegated authentication protocols (**MEDAP-II** and **MEDAP-III**) among multiple mobile data owners and multiple cloud servers, based on our IBMPMS scheme.
- Finally, **MEDAPs** support a conditional delegated authentication that is according to the security model, any compromised cloud servers and mobile data owners could lead to authentication services ceased.

The organization of the rest paper is as follows. Section 2 introduces our problem formalized. Section 3 and Section 4 give the necessary mathematic preliminaries and formal definition of IBMPS scheme in cryptograhy, respectively. In Section 5, we propose a conctete IBMPS construction in cryptogaphy proven secure in Section 6. Section 7 gives the detailed application of IBMPS scheme in a delegated authentication protocols. Section 8 also extends our IBMPS and designs a novel delegated authentication protocols supporting multiple data owners. Section 9 lists some related work. Section 10 concludes the paper at the end of this paper.

2. SYSTEM FORMULATION

We present the system architectures, security problems, and design goals.

2.1. System Architectures

we have considered a mobile cloud computing model which is illustrated in Figure 1. In this model, it is constituted of four entities: *system authority*, *cloud servers*, *mobile data owners* and *mobile data consumers*.

- **System Authority (SA).** SA administrates the whole system, which is independent with other entities in system. SA is in charge of the system setup and key distribution to the participants. It is assumed that SA controlled by the government can not be broken down by any adversaries.

- **Cloud Servers.** The core of the cloud consists of a group of servers, denoted as $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n$, with computation and storage resources, as a general cloud computing model. These cloud servers process huge of computation and storage capability compared with the data owners and data consumers.
- **Mobile Data Owners.** It is assumed that the data owners work in a mobile way in our model. The mobile data owners may be the mobile sensors and upload to the cloud when the mobile data owners eventually encounter to the networks. As a result, the mobile data owners can not be always available. From the security consideration, the mobile data owners can add authenticated tags with the data achieved by digital signature.
- **Data Consumers.** In our model, the data consumers always request the data and computation due to their lower computation and smaller storage capability when they demand. From the security aspect, the data consumers want to fetch the authenticated data and computing results to keep the authenticity, which is hard to achieve when the data owner is unavailable.

2.2. Threat Models

In this work, it is assumed that the adversary can get all the identities in the system. We also consider an extreme situation that the attackers control the most participants and work against only one honest signer. In addition, we assume that there exist no secure channels between the mobile data owners and cloud servers. According to the different ability of the attacker, the threat model can be classified into two catalogues.

- The adversary could compromise both of the mobile data owner and at most $n-1$ cloud servers. As a result, the attacker has obtained the secret keys of the mobile data owner and $n-1$ cloud servers except *one* cloud server. The adversary aims to make a valid multi-signature and provide data authentication services for cloud users without the participation of the honest cloud servers.
- The adversary could compromise all of the n cloud servers. In this case, the adversary knows the secret keys of n cloud servers except the data owner. The adversary aims to make a valid delegation from the data owner and further make a valid multi-signature to provide data authentication services for cloud users without the delegation of the data owner.

2.3. Design Goals

The protocols are to achieve the following goals.

- **Data authentication.** The cloud consumers could obtain the authenticated data from the cloud as well as the computational results.
- **Delegated authentication.** The mobile data owners could conditionally authorize the cloud servers without directly leaking the signing keys.
- **Multiple entities.** The data owners and cloud servers could be distributed corresponding to different situations and preventing single point of failure.
- **Efficiency.** The computational overhead is lower than other bilinear pairing based protocols and the communication overhead does not linear grow with the number of cloud servers.

3. MATHEMATICAL PRELIMINARIES

3.1. Cubic Residue

Firstly, we give an introduction of *cubic residue* [15].

Definition 1 (cubic residue)
For an integer $N > 0$, $a \in \mathbb{Z}_N^*$ is a *cubic residue* modulo N if there exists some integer $x \in \mathbb{Z}_N^*$ which satisfies $x^3 \equiv a \pmod{N}$. x is named as a *cubic root* of a modulo N.

DOI: 10.1002/sec
Prepared using secauth.cls

After that, we introduce the following lemma [15, 16] about cubic residue.

Lemma 1 (cubic residue construction)
If p is a prime with $p \equiv 2 \pmod 3$ and q is also a prime with $q \equiv 4$ or $7 \pmod 9$, we can produce a cubic residue modulo N. Let a be a non-cubic modulo q, for any $h \in \mathbb{Z}_N^*$ we can compute that

$$\eta = \frac{(q-1) \pmod 9}{3}, \quad \lambda = \eta \pmod 2 + 1, \quad \beta = (q-1)/3, \quad \xi \equiv a^{\eta\beta} \pmod q, \quad \tau \equiv h^{\lambda\beta} \pmod q$$

and

$$b = \begin{cases} 0, & \text{if } \tau = 1 \\ 1, & \text{if } \tau = \xi \\ 2, & \text{if } \tau = \xi^2 \end{cases}$$

we can construct a cubic residue modulo N that is $C = a^b \cdot h \pmod N$.

Theorem 1
Let p, q, N, C, and η be defined in *Lemma* 1, we can get a cubic root of C^{-1} by $s \equiv C^{[2^{\eta-1}(p-1)(q-1)-3]/9} \pmod N$. Note that $s^3 \cdot C \equiv 1 \pmod N$.

3.2. Integer Factorization Assumption

let $N = pq$, where p, q are large primes. The integer factorization problem is easier if we can seek out two different cubic roots s_1, s_2 of cubic residue a modulo N, where $s_1^3 \equiv s_2^3 \equiv a \pmod N$ and $s_1 \neq \pm s_2 \pmod N$. Thus, we can factor N by executing Algorithm $Fac(a, s_1, s_2)$ in polynomial time [17].

Algorithm 1 ($Fac(a, s_1, s_2)$)
If $s_1 \neq \pm s_2 \pmod N$, $\gcd(s_1 + s_2, N)$ or $\gcd(s_1 - s_2, N)$ as non-trivial divisor of N, where $\gcd(,)$ is an algorithm to find out greatest common divisor.

4. CRYPTOGRAPHIC DEFINITIONS

4.1. Definition of IBMPS scheme

Let $\mathcal{O}$ be the original signer (the data owner in the mobile cloud model) and $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ be a group of proxy signers (the cloud servers in the mobile cloud model) designated by $\mathcal{O}$. In addition, we denote that the identity of $\mathcal{O}$ and $\mathcal{P}_i$ are as $ID_{\mathcal{O}}$ and $ID_{\mathcal{P}_i}$, respectively.

Definition 2 (Framework)
An identity-based multi-proxy signature scheme is defined as a following *seven* tuple **IBMPS=(Setup, Extra, Sign, Verify, MPGen, MPSign, MPVerify)** [18].

- **Setup**(1^k): The **PKG** runs this algorithm by taking a security parameter k as input, and outputs the whole system sharing public parameters *pp* and master secret keys *msk* which **PKG** keeps in secret.
- **Extra**(ID, msk, pp): The algorithm is run by **PKG** on input a participant's identity denoted as ID, master secret keys msk **PKG** holds, and public parameters pp and generates the corresponding private key d_{ID}. Thus, **PKG** uses this algorithm to generate secret keys for all participators in the system and distributes through some secure channels.
- **Sign**(ID, d_{ID}, m, pp): This algorithm takes the signers identity ID and its secret key d_{ID}, the data to be signed m and pp as input, outputs a signature tag σ on the data m.

- **Verify**(ID, m, σ, pp): This algorithm is run by any verifiers by taking the signers identity ID, the data m, a candidate signature tag σ and pp as input. If σ is a valid signature tag, this algorithm outputs 1. Otherwise, it outputs 0.
- **MPGen**($ID_{\mathcal{O}}$, $ID_{\mathcal{P}_1}$, $ID_{\mathcal{P}_2}$, ..., $ID_{\mathcal{P}_n}$, $d_{\mathcal{O}}$, $d_{\mathcal{P}_1}$, $d_{\mathcal{P}_2}$, ..., $d_{\mathcal{P}_n}$, ω, pp): This is an interactive process among the original signer and the designated proxy signers. All of the participants take as input their identities $ID_{\mathcal{O}}$, $ID_{\mathcal{P}_1}$, $ID_{\mathcal{P}_2}$, ..., $ID_{\mathcal{P}_n}$ and their secret keys $d_{\mathcal{P}_1}$, $d_{\mathcal{P}_2}$, ... , $d_{\mathcal{P}_n}$ and the delegation warrant ω. As a result, each proxy signer gets a partial proxy signing key $sk_{\mathcal{P}_i}$ which is used to cooperatively produce IBMPS by the proxy signers instead of the original signers.
- **MPSign**($sk_{\mathcal{P}_i}$, ω, m, pp): Each proxy signer takes its proxy signing key $sk_{\mathcal{P}_i}$, the delegation warrant ω, and the data m as input. The algorithm outputs an IBMPS $p\sigma$ on behalf of $\mathcal{O}$.
- **MPVerify**($ID_{\mathcal{O}}$, $ID_{\mathcal{P}_1}, ID_{\mathcal{P}_2}, ..., ID_{\mathcal{P}_n}$, ω, m, $p\sigma$, pp): This algorithm takes the every participant's identity $ID_{\mathcal{O}}, ID_{\mathcal{P}_1}, ID_{\mathcal{P}_2}, ..., ID_{\mathcal{P}_n}$, the delegation warrant ω, the data m and a candidate multi-proxy signature tag $p\sigma$. Finally, if $p\sigma$ is indeed a valid IBMPS, it outputs 1. Otherwise, it outputs 0.

4.2. Security Model in Cryptography

We assume that the attacker $\mathcal{A}$ has known all the identities of original signer and proxy signers. $\mathcal{A}$ plays the role of the original signer or one or more of the proxy signers and requests to other honest participants. $\mathcal{A}$ can be divided into two catalogues:

- $\mathcal{A}_1$. The adversary has compromised the original signer $\mathcal{O}$ and $n-1$ proxy signers and obtained their secret keys. In the case, $\mathcal{A}_1$ can output a valid delegation from $\mathcal{O}$ to the proxy signers. The object of $\mathcal{A}_1$ is to forge a valid multi-proxy signature under the valid delegation by n proxy signers.
- $\mathcal{A}_2$. The adversary has compromised n proxy signers and obtained their secret keys except the original signer $\mathcal{O}$. In the case, $\mathcal{A}_2$ can output a valid multi-signature by n proxy signers. The object of $\mathcal{A}_2$ is to forge a valid delegation from $\mathcal{O}$.

Definition 3
Consider the games between $\mathcal{A}_i$ $(i = 1 \; or \; 2)$ and the challenger $\mathcal{C}$:

Step 1 $\mathcal{C}$ runs **Setup** and gets the public parameters pp and the master secret key msk. The attacker $\mathcal{A}$ is sent pp while $\mathcal{C}$ keeps msk.

Step 2 $\mathcal{C}$ controls *five* oracles: a hash oracle $\mathcal{O}_{hash}$, a key extract oracle $\mathcal{O}_{ext}$, a standard signing oracle $\mathcal{O}_{sign}$, a delegation oracle $\mathcal{O}_{del}$, and a multi-proxy signature oracle $\mathcal{O}_{mps}$ which are used to provide a real distinguished environment for the adversary.

Step 3 $\mathcal{A}_i$ adaptively queries to $\mathcal{C}$ as follows.

- **Extra queries.** When $\mathcal{A}_i$ asks for the secret key of signer's identity ID, $\mathcal{C}$ returns the secret key d_{ID} to $\mathcal{A}$ by running algorithm **Extra** and then puts the tuple (ID, c, d_{ID}) into a list $list_E$.
- **Sign queries.** When $\mathcal{A}$ can query the signature under the identity ID on data m, $\mathcal{C}$ visits the standard signing oracle $\mathcal{O}_{sign}$ and returns a standard signature σ for m. Then tuple (ID, σ, m) is added to a list $list_S$.
- **Delegation queries.** When $\mathcal{A}$ requests a delegation on proxy signer ID_i under warrant ω, $\mathcal{C}$ returns the delegation σ and puts the tuple (ID_i, ω, σ) into a list $list_W$.
- **MPSign queries.** When $\mathcal{A}$ requests multi-proxy signature on m under the warrant ω, $\mathcal{C}$ outputs a partial proxy signature $p\sigma$ on data m. The tuple $(\omega, m, p\sigma)$ is added to a list $list_M$.

Step 4 Eventually, $\mathcal{A}$ outputs a forgery by winning the above games if one of the events satisfies.

1. $\mathcal{A}_1$'s goal is to make the forgeries.

DOI: 10.1002/sec
Prepared using secauth.cls

- A correct standard signature (m^*, σ^*) by an honest signer ID_n on a data m^* was not queried to $\mathcal{O}_{sign}$ but $Verify(ID_n, m^*, \sigma^*) = 1$.
- A correct multi-proxy signature $(\omega^*, m^*, p\sigma^*)$ on a data m^* under the warrant ω^* by an honest signer ID_n on behalf of the original signer such that the original signer never authorize the only honest signer that is $(ID_i^*, \omega^*) \notin list_W$ and $(\omega^*, m^*) \notin list_M$.

2. $\mathcal{A}_2$'s goal is to make the forgeries:
   - A correct standard signature (m^*, σ^*) by the honest signer ID_n on a data m^* was not queried to $\mathcal{O}_{sign}$ and $Verify(ID_n, m^*, \sigma^*) = 1$.
   - A correct multi-proxy signature $(\omega^*, m^*, p\sigma^*)$ on a data m^* under the warrant ω^* where $(\omega^*, m^*) \notin list_M$, by any proxy signer ID_i which was never authorized by the original signers that is $(ID_i^*, \omega^*) \notin list_W$.

We denote that the advantage of adversary $\mathcal{A}_i$, $Adv_{IBMPS}^{\mathcal{A}_i}$ is the $\mathcal{A}_i$'s probability of success in the above game.

Definition 4
For any adversary $\mathcal{A}_i$, if $Adv_{IBMPS}^{\mathcal{A}_i}$ is negligible, the IBMPS scheme is secure against chosen message attacks and chosen identity/warrant attacks.

5. IBMPS SCHEME CONSTRUCTION

We construct a novel IBMPS scheme based on the Wang's identity-based signature scheme [15]. As we defined in Section 4, our scheme consists of *seven* algorithms: *Setup*, *Extra*, *Sign*, *Verify*, *MPGen*, *MPSign*, and *MPVerify*.

5.1. Setup

By taking the parameter k, the algorithm **Setup** is run by **PKG**.

1) **PKG** randomly chooses two primes p and q where p satisfies $p \equiv 2 \pmod 3$ and q satisfies $q \equiv 4 \pmod 9$ or $q \equiv 7 \pmod 9$. After that, **PKG** computes the module $N = p \cdot q$.
2) Following the Lemma 1, **PKG** sets η, λ, β which satisfy $\eta = [q - 1 \pmod 9]/3$, $\lambda = \eta \pmod 2 + 1$, and $\beta = (q-1)/3$.
3) **PKG** also chooses a non-cubic residue a such that $\left(\frac{a}{q}\right) = -1$ and sets $\xi = a^{\eta\beta} \pmod q$.
4) **PKG** picks up *five* hash functions H_1, H_2, H_3, H_4, and H_5, in which $H_1, H_4 : \{0,1\}^* \to \mathbb{Z}_N^*$, $H_2, H_3, H_5 : \{0,1\}^* \to \{0,1\}^\ell$.

PKG obtains master secret keys $msk = (p, q, \beta)$ and public parameter $pp = (N, H_1, H_2, H_3, H_4, H_5, a, \eta, \lambda)$.

5.2. Extra

For each users, by taking an identity ID, **PKG** extracts the secret key d_{ID}.

1) **PKG** computes $\omega = H_1(ID)^{\lambda\beta} \pmod q$.
2) **PKG** sets a flag $c = \begin{cases} 0, & \text{if } \omega = 1 \\ 1, & \text{if } \omega = \xi \\ 2, & \text{if } \omega = \xi^2 \end{cases}$ and sets a general function $H(ID) = a^c \cdot H_1(ID) \pmod N$. Note that $H(ID) \in \mathbb{CR}_N$.
3) **PKG** generates the secret key by

$$d_{ID} = H(ID)^{\frac{2^{\eta-1}(p-1)(q-1)-3}{9}} \pmod N, \tag{1}$$

PKG distributes d_{ID} along a flag c in a secure way. Everyone in the system can compute such a value $H(ID)$ according to public ID and c.

5.3. Sign

The original signer $\mathcal{O}$ generates the signature on the data m.

1) $\mathcal{O}$ randomly chooses an integer $r \in \mathbb{Z}_N^*$ and computes its cube by $R \equiv r^3 \pmod N$.
2) $\mathcal{O}$ sets the hash value $h_2 = H_2(m||R)$ and then computes the signature by $V \equiv r \cdot d_{\mathcal{O}}^{h_2} \pmod N$.

The signature is $\sigma = (R, V)$.

5.4. Verify

Anyone could check the signature σ on data m by phasing σ to (R, V) and checking Equation (2)

$$V^3 \cdot H(ID_{\mathcal{O}})^{h_2} \stackrel{?}{=} R, \tag{2}$$

where $h_2 = H_2(m||R)$.

5.5. MPGen

To authorize the proxy signers $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$, the original signer $\mathcal{O}$ makes the delegation warrant ω, which specifies necessary delegation details, including the delegation types, the legal range of the proxy servers, and the expiry time of delegation, etc. $\mathcal{P}_i$ generates a new signing key of proxy signers in an interactive way as follows.

5.5.1. Delegation generation.

The original signer $\mathcal{O}$ first confirms the delegation warrant ω by signing on it.

1) $\mathcal{O}$ randomly chooses an integer $r_o \in \mathbb{Z}_N^*$ and computes its cube $R_o = r_o^3$.
2) $\mathcal{O}$ sets the hash value $h_2 = H_2(\omega||R_o)$ and computes $V_o = r_o \cdot d_o^{h_2}$.
3) $\mathcal{O}$ further broadcasts the delegation (ω, R_o, V_o) to the designated proxy signers $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$.

5.5.2. Delegation verification.

Once receiving delegation requests, each proxy signer $\mathcal{P}_i$ first confirms ω and checks Equation (3)

$$V_o^3 \cdot H(ID_o)^{h_2} \stackrel{?}{=} R_o, \tag{3}$$

where $h_2 = H_2(\omega||R_o)$. Proxy signers accept the delegation if Equation (3) is satisfied. Otherwise, it aborts.

5.5.3. Proxy key generation.

Once accepting the delegations, each proxy signer $\mathcal{P}_i$ generates its new signing key by

$$sk_{p_i} = V_o \cdot d_{p_i}^{h_3}, \tag{4}$$

where $h_3 = H_3(\omega||R_o)$. We suggest using H_3 instead of H_2 for security requirement. sk_{p_i} is used to partially sign the data on behalf of $\mathcal{O}$.

5.6. MPSign

Each proxy signer $\mathcal{P}_i$ signs the data m under ω cooperatively on behalf of $\mathcal{O}$.

DOI: 10.1002/sec
Prepared using secauth.cls

1) $\mathcal{P}_i$ randomly chooses an integer $r_i \in \mathbb{Z}_N$ and computes its cube $R_i = r_i^3 \pmod N$. $\mathcal{P}_i$ computes $t_i = H_4(R_i)$ and broadcasts t_i to other cloud servers $\mathcal{P}_j$.
2) Once receiving t_j, $\mathcal{P}_i$ broadcasts R_i to other proxy signers.
3) Once receiving R_j, $\mathcal{P}_i$ checks $t_j \stackrel{?}{=} H_4(R_j)$. If one of equations dissatisfies, aborts. Otherwise, $\mathcal{P}_i$ computes $U = \prod_{i=1}^{n} R_i \pmod N$ and sets the hash $h_5 = H_5(\omega||m||U)$ and computes $v_i = r_i \cdot sk_{p_i}^{h_5} \pmod N$ and broadcasts v_i to other proxy signers.
4) Once receiving v_j from other proxy signers, $\mathcal{P}_i$ first checks each Equation (i=1,2,...,n)

$$v_i^3 \cdot H(ID_{p_i})^{h_3 h_5} \stackrel{?}{=} R_i \cdot V_o^{h_5}, \tag{5}$$

where $h_3 = H_3(\omega||R_o)$ and $h_5 = H_5(\omega||m||U)$. If one of the equations dissatisfies, it aborts. Otherwise, $\mathcal{P}_i$ computes $V = \prod_{i=1}^{n} v_i \pmod N$.

Once all partial signatures are correct, the IBMPS on data m can be aggregated as $p\sigma = (\omega, R_o, V_o, U, V)$

5.7. MPVerify

Once receiving the data m and the IBMPS $p\sigma = (\omega, R_o, V_o, U, V)$, any verifier operates the following verification algorithm.

1) The verifier checks whether m conforms to ω.
2) The verifier checks whether $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ are authorized by $\mathcal{O}$ according to the warrant ω.
3) The verifier accepts IBMPS if and only each of Equation (6) holds:

$$V^3 \cdot H(ID_o)^{h_2 h_5} \cdot \prod_{i=1}^{n} H(ID_{p_i})^{h_3 h_5} \stackrel{?}{=} U \cdot V_o^{h_5} \tag{6}$$

where $h_3 = H_3(\omega||R_o)$ and $h_5 = H_5(\omega||m||U)$.

6. SECURITY ANALYSIS

6.1. Correctness

The IBMPS scheme is correct since

$$\begin{aligned} V &\equiv \prod_{i=1}^{n} v_i \equiv \prod_{i=1}^{n} r_i \cdot sk_{p_i}^{h_5} \equiv \prod_{i=1}^{n} r_i \cdot (V_o \cdot d_{p_i}^{h_3})^{h_5} \\ &\equiv \prod_{i=1}^{n} r_i \cdot (r_o \cdot d_o^{h_2})^{h_5} \cdot d_{p_i}^{h_3 h_5} \equiv \prod_{i=1}^{n} r_i \cdot r_o^{h_5} \cdot d_o^{h_2 h_5} \cdot d_{p_i}^{h_3 h_5} \pmod N, \end{aligned} \tag{7}$$

we have

$$\begin{aligned} V^3 &\equiv \prod_{i=1}^{n} R_i \cdot V_o^{h_5} \cdot (d_o^3)^{h_2 h_5} \cdot (d_{p_i}^3)^{h_3 h_5} \equiv U \cdot V_o^{h_5} \cdot (d_o^3)^{h_2 h_5} \cdot \prod_{i=1}^{n} (d_{p_i}^3)^{h_3 h_5} \\ &\equiv U \cdot V_o^{h_5} \cdot (H(ID_o)^{-1})^{h_2 h_5} \cdot \prod_{i=1}^{n} (H(ID_{p_i})^{-1})^{h_3 h_5} \pmod N. \end{aligned} \tag{8}$$

Finally, we have finished it since

$$V^3 \cdot H(ID_o)^{h_2 h_5} \cdot \prod_{i=1}^{n} H(ID_{p_i})^{h_3 h_5} \equiv R \cdot V_o^{h_5} \pmod{N}. \quad (9)$$

6.2. Security Proof

Theorem 2 (Main Theorem)
If the integer factoring problem is (t', ϵ')-hard, our IBMPS scheme is $(t, q_E, q_S, q_H, n, \epsilon)$-secure against existential forgery on the adaptively chosen message attack and chosen identity/warrant attack which is satisfying

$$\epsilon' \geq \frac{4}{9}\Big(\frac{\epsilon^2}{2(q_H+1)} - \frac{(2n+6)q_S q_H}{2^N}\Big) \quad (10)$$

$$t' \geq 2t + O(k^2\ell + k^3) \quad (11)$$

where k and ℓ are security parameters.

Proof
It is assumed that there is an attacker $\mathcal{A}$ which has the ability to break our IBMPS scheme, we could construct an algorithm $\mathcal{C}$ as a simulator which plays with the adversary $\mathcal{A}$ and solves the integer factorization problem at the end of the game.

Given an instance N for two unknown primes p and q product, our final target is that the simulator $\mathcal{C}$ could output p or q at the end of the simulation.

The simulator $\mathcal{C}$ and adversary $\mathcal{A}$ take part in the following games. $\mathcal{C}$ sends the public parameter $pp = \{N, H_1, H_2, H_3, H_4, H_5, a, \eta, \lambda\}$ to $\mathcal{A}$. $\mathcal{C}$ maintains several lists ($list_{h_1}$, $list_{h_2}$, $list_{h_3}$, $list_{h_4}$, $list_{h_5}$, $list_S$, $list_W$, $list_M$, $list_{MPS}$) which are empty in the beginning. $\mathcal{C}$ responses $\mathcal{A}$'s requests as follows.

- H_1**-query**. To response to $\mathcal{A}$'s query, $\mathcal{C}$ maintains a list $list_{h_1}$ with the tuples (ID, c, h_1, s). Once a query on ID, $\mathcal{C}$ first checks $list_{h_1}$ and returns h_1 if the ID already exists in $list_{h_1}$. Otherwise, $\mathcal{C}$ randomly picks up two integers $s \in \mathbb{Z}_N$ and $c \in \{0, 1, 2\}$, respectively, and sets $h_1 = H(ID) = s^3/a^c \pmod{N}$ $\mathcal{C}$, returns h_1 as an answer to $\mathcal{A}$ and adds the entry (ID, c, h_1, s) to $list_{h_1}$.
- **Extra-query**. $\mathcal{C}$ queries ID in $list_{h_1}$ and actively performs a H_1-query on ID if ID is not yet defined. $\mathcal{C}$ returns s as a secret key to $\mathcal{A}$.
- H_2**-query**. $\mathcal{C}$ sets up $list_{h_2}$ with the tuples (m, R, h_2). Once $\mathcal{A}$ requests hash function H_2 on (m, R), $\mathcal{C}$ returns h_2 directly if the tuple (m, R) is in $list_{h_2}$. Otherwise, $\mathcal{C}$ randomly chooses $h_2 \in \{0, 1\}^\ell$ and adds (m, R, h_2) into $list_{h_2}$ and returns h_2 to $\mathcal{A}$.
- H_4**-query**. $\mathcal{C}$ establishes a list $list_{h_4}$ with the tuples (R, h_4). Once a query on the randomness R, $\mathcal{C}$ returns h_4 if the R in the list. Otherwise, $\mathcal{C}$ randomly chooses h_4 and adds (R, h_4) into $list_{h_4}$ and returns h_4 to $\mathcal{A}$.
- H_5**-query**. $\mathcal{C}$ maintains a list $list_{h_5}$ with the tuples (ω, m, R, h_5). Once $\mathcal{A}$ requests hash function H_5 on (ω, m, R), $\mathcal{C}$ directly returns h_5 if the tuple (ω, m, R) is in the list $list_{h_5}$. Otherwise, $\mathcal{C}$ randomly chooses $h_5 \in \{0, 1\}^\ell$ and adds (ω, m, R, h_5) into $list_{h_5}$ and returns h_5 to $\mathcal{A}$.
- **Sign-query**. Once $\mathcal{A}$ queries a signature by ID on m, $\mathcal{C}$ first computes $H(ID) = a^c \cdot h_1$ since $\mathcal{C}$ controls $list_{h_1}$, and randomly picks two integers $V \in \mathbb{Z}_N$ and $h_2 \in \{0, 1\}^\ell$. If the tuple $(m, V^3 \cdot H(ID)^{h_2})$ is not in the $list_{h_2}$, $\mathcal{C}$ sets $R = V^3 \cdot H(ID)^{h_2})$ and returns (R, V) as the signature to $\mathcal{A}$. $\mathcal{C}$ then adds (ID, m, R, V) to $list_S$, and adds (m, R, h_2) to $list_{h_2}$. In short, $\mathcal{C}$ can control the signature oracle and play with $\mathcal{A}$ to answer any signature even if $\mathcal{C}$ does not obtain the secret key.
- Delegation-**query**.
  - When $\mathcal{A}_1$ requests one of the honest proxy signers, we denote proxy singer with identity ID_n. $\mathcal{A}_1$ chooses a warrant ω and generates the signature $\sigma = (Ro, Vo)$ for ω. Then $\mathcal{A}_1$ sends (ω, σ) to $\mathcal{C}$. Verifying the validity of σ, $\mathcal{C}$ adds ω to $list_W$.

DOI: 10.1002/sec
Prepared using secauth.cls

– If $\mathcal{A}_2$ requests to interact with the original signer, $\mathcal{A}$ chooses a warrant ω, and asks original signer to sign on ω. In this case, due to the lack of the secret key, $\mathcal{C}$ queries to the standard signing oracle $\mathcal{O}(d_o, \cdot)$ for help. Once getting a result σ from $\mathcal{O}$, $\mathcal{C}$ sends (ω, σ) to $\mathcal{A}_2$ and adds ω to $list_W$.

- **MPSign-query**. $\mathcal{A}$ asks a multi-proxy signature on $(p\sigma, m)$, in which the user ID_1 is in the proxy group. $\mathcal{C}$ returns in the following ways.

  1. When ω does not exist in the $list_W$, $\mathcal{C}$ must halt at once. Otherwise, $\mathcal{C}$ gets $\sigma_o = (R_o, V_o)$ through the standard signature oracle.
  2. $\mathcal{C}$ recovers $H_1(ID_i)$ from the $list_{h_1}$.
  3. $p\sigma = (m, \omega, R_0, R, V)$ is a valid proxy signature on m by ID_i $(i = 1, 2, ..., n)$ on behalf of ID_0 under ω.
  4. $\mathcal{C}$ adds the tuple (ω, m) to $list_M$, and sends $p\sigma$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ either fails or returns a forgery.

We apply the Forking Lemma [19] by replaying the oracle technique. Since $\mathcal{C}$ controls all the oracles, $\mathcal{C}$ resets $\mathcal{A}$ twice. In the first time, $\mathcal{C}$ records all the transcripts with $\mathcal{A}$. In the second time, $\mathcal{C}$ returns all the same value from the transcripts except H_2 oracle. $\mathcal{C}$ returns two random integers t_1, t_2 as the answer, respectively.

1. $\mathcal{A}$ forges a signature (R^*, V^*) on m^* and ID, where $(ID, m^*, R^*, V^*) \notin list_S$, and

$$(V^*)^3 \cdot H(ID)^{h_2} \equiv R^* \pmod{N}.$$

   According to the Forking Lemma, $\mathcal{A}$ could forge two signatures (R^*, V^*) and (R^*, V^{**}) on m^* and ID and sends to $\mathcal{C}$. $\mathcal{C}$ looks up $list_{h_2}$ to obtains t_1 and t_2.
   If $t_1 - t_2 \not\equiv 0 \pmod 3$, $\mathcal{C}$ could get $(V^*)^3 S^{t_1} \equiv R^* \pmod N$ and $(V^{**})^3 S^{t_2} \equiv R^* \pmod N$, where $S = a^c \cdot H(ID) = s^3$ for unknown s. $\mathcal{C}$ can get

$$(V^*/V^{**})^3 \equiv S^{t_2 - t_1} \pmod{N}.$$

   It can be divided into two cases:

   - If $t_2 - t_1 \equiv 1 \pmod 3$, for some integer k, we have $t_2 - t_1 = 3k + 1$. Therefore, $S \equiv (\frac{V^*}{V^{**} S^k})^3$, that is $s = \frac{V^*}{V^{**} S^k}$.
   - If $t_2 - t_1 \equiv -1 \pmod 3$, for some integer k, we have $t_2 - t_1 = 3k - 1$. Therefore, $S \equiv (\frac{V^* S^k}{V^{**}})^3$, that is $s = \frac{V^* S^k}{V^{**}}$.

   Therefore, $\mathcal{C}$ can get a cubic root s of S. Furthermore, if $\mathcal{C}$ gets two different cubic root of S, $\mathcal{C}$ can factor N Through *Algorithm* 1.
2. When $\mathcal{A}_1$ forges a signature $(\omega^*, R^*, V_o^*, V^*)$ on m^*, where user $ID_{\mathcal{O}}$ is the original signer, $\omega^* \notin list_W$, and

$$V^3 \cdot H(ID_{\mathcal{O}})^{h_2 h_5} \cdot \prod_{i=1}^{n} H(ID_{p_i})^{h_3 h_5} \stackrel{?}{=} R \cdot V_o^{h_5}$$

   where $h_3 = H_3(\omega || R_o)$ and $h_5 = H_5(\omega || m || R)$.
3. When $\mathcal{A}_2$ forges a signature $(\omega^*, R^*, V_o^*, V^*)$ on m^*, where $(\omega^*, m^*) \notin list_{mps}$, and

$$V^3 \cdot H(ID_o)^{h_2 h_5} \cdot \prod_{i=1}^{n} H(ID_{p_i})^{h_3 h_5} \stackrel{?}{=} R \cdot V_o^{h_5}$$

   where $h_3 = H_3(\omega || R_o)$ and $h_5 = H_5(\omega || m || R)$.

Thus, we have finished the proof. □

7. MEDAP-I: THE APPLICATION OF IBMPS SCHEME

7.1. Multi-Entities Delegated Authentication Protocol in Cloud

We build a multi-entities delegated authentication protocol **MEDAP-I** in [20] using our novel IBMPS scheme. **MEDAP-I** is made up of *five* steps: *System initialization*, *Authentication tag*, *Authentication delegation*, *Multi-authentication*, *Verification*.

7.1.1. System initialization

This step sets up the system parameters and distributes every entity's secret key. **SA** runs **Setup** algorithm by taking the parameter k and obtains $msk = (p, q, \beta)$ and $pp = (N, H_1, H_2, H_3, H_4, H_5, a, \eta, \lambda)$. For every participant with identity ID, **SA** runs **Extra** algorithm and generates the corresponding secret key d_{ID} for the entity.

7.1.2. Authentication tags

- **Tag generation.** The mobile data owner $\mathcal{O}$ computes an authentication tag for the data m by using **Sign** algorithm. $\mathcal{O}$ randomly chooses an integer r where $r \in \mathbb{Z}_N^*$ and then computes its cube $R = r^3 \pmod N$. $\mathcal{O}$ sets $h_2 = H_2(m||R)$ and computes $V \equiv r \cdot d_{\mathcal{O}}^{h_2} \pmod N$. The authentication tag is $tag = (R, V)$ which is along with the data m and uploaded to the cloud servers.
- **Tag verification.** Once receiving the data m and its corresponding authentication tag tag, the cloud servers verify tag by using the **Verify** algorithm and check $V^3 \cdot H(ID_{\mathcal{O}})^{h_2} \stackrel{?}{=} R$ where $h_2 = H_2(m||R)$. The cloud confirms m if tag is a valid one. Otherwise, it aborts.

7.1.3. Authentication delegation

To authorize the authentication rights to mobile cloud, $\mathcal{O}$ makes the valid delegated warrant ω in *three* steps.

- **Delegation generation.** $\mathcal{O}$ firstly confirms ω and signs on it. $\mathcal{O}$ randomly chooses an integer $r_o \in \mathbb{Z}_N^*$, computes its cube $R_o = r_o^3$, sets $h_2 = H_2(\omega||R_o)$ and computes $V_o = r_o \cdot d_{\mathcal{O}}^{h_2}$. $\mathcal{O}$ broadcasts $\sigma = (\omega, R_o, V_o)$ to the designated cloud servers $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$.
- **Delegation verification.** Once receiving the requests from $\mathcal{O}$, every $\mathcal{P}_i$ firstly confirms $\sigma = (\omega, R_o, V_o)$ by checking $V_o^3 \cdot H(ID_{\mathcal{O}})^{h_2} \stackrel{?}{=} R_o$ where $h_2 = H_2(\omega||R_o)$. $\mathcal{P}_i$ accepts σ if valid. Otherwise, $\mathcal{P}_i$ requests a new delegation from $\mathcal{O}$, or aborts.
- **Proxy key generation.** Once accepting the requests, $\mathcal{P}_i$ generates a new proxy singer key by $sk_{p_i} = V_o \cdot d_{p_i}^{h_3}$, where $h_3 = H_3(\omega||R_o)$.

7.1.4. Multi-authentication tag

Cloud servers $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ cooperate to sign on the data m under the warrant ω on behalf of the mobile data owner $\mathcal{O}$ in an interactive way.

- Each $\mathcal{P}_i$ chooses a random integer $r_i \in \mathbb{Z}_N$ and computes its cube $R_i = r_i^3$ and $t_i = H_4(R_i)$ and broadcasts t_i.
- Once receiving t_j $(j \neq i)$, $\mathcal{P}_i$ broadcasts R_i to other servers.
- Once receiving R_j $(j \neq i)$, $\mathcal{P}_i$ checks each $t_j \stackrel{?}{=} H_4(R_j)$. If one of the equations dissatisfies, it aborts the protocol. $\mathcal{P}_i$ aggregates $R \equiv \prod_{i=1}^{n} R_i \pmod N$, sets $h_5 = H_5(\omega||m||R)$, computes $v_i \equiv r_i \cdot sk_{p_i}^{h_5} \pmod N$, and broadcasts v_i.
- Once receiving v_j $(j \neq i)$, $\mathcal{P}_i$ checks each equation $v_j^3 \cdot H(ID_{\mathcal{P}_j})^{h_3 h_5} \stackrel{?}{=} R_j \cdot V_o^{h_5}$, where $h_3 = H_3(\omega||R_o)$ and $h_5 = H_5(\omega||m||R)$. If one of the equations dissatisfies, the protocol aborts immediately. $\mathcal{P}_i$ aggregates $V \equiv \prod_{i=1}^{n} v_i \pmod N$.

Once all the designated cloud servers have finish the above algorithm correctly, the multi-authentication tag $p\sigma$ of the data m can be generated by all the cloud servers as $p\sigma = (\omega, R_o, V_o, R, V)$

DOI: 10.1002/sec
Prepared using secauth.cls

7.1.5. Authentication verification

Once receiving m and the multi-authentication tag $p\sigma = (\omega, R_o, V_o, R, V)$, the data consumer does the following operations.

- The data consumers checks that $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ are legally authorized by $\mathcal{O}$ according to ω.
- The data consumers accepts $p\sigma$ if and only if $V^3 \cdot H(ID_{\mathcal{O}})^{h_2 h_5} \cdot \prod_{i=1}^{n} H(ID_{\mathcal{P}_i})^{h_3 h_5} \stackrel{?}{=} R \cdot V_o^{h_5}$ where $h_3 = H_3(\omega||R_o)$ and $h_5 = H_5(\omega||m||R)$.

Finally, the data consumer is convinced to accept the data which are authenticated by the mobile data owner $\mathcal{O}$. The **MEDAP-I** finishes.

8. MEDAP-II: EXTENSION PROTOCOLS

In some scenarios, a certain scope of the mobile data owners are sharing one copy of data that all of these data owners are responsible for. In this case, all of these data owners need to sign on the data and further aggregate to one signature, which is named as multi-signature. When it comes to the authentication delegation, all of these mobile data owners interactive to negotiate a common delegation credential to the cloud servers which play the role of proxy signers. In this section, we extend single data owner in our **MEDAP-I** in Section 7 to multiple data owners, shown in Figure 2. We design a novel protocol built on a cryptogrphic primitive, an identity-based multi-proxy multi-signature (IBMPMS). According to our IBMPS scheme, we can continue to construct IBMPMS scheme through a transforming one original signer to multiple original signers. In addition, when the cloud servers are considered into one entity, through the same employment, we also design a protocol **MEDAP-III** built on Wang's work an identity-based proxy multi-signature (IBPMS) [16] to adapt to the different demands in the mobile cloud computing.

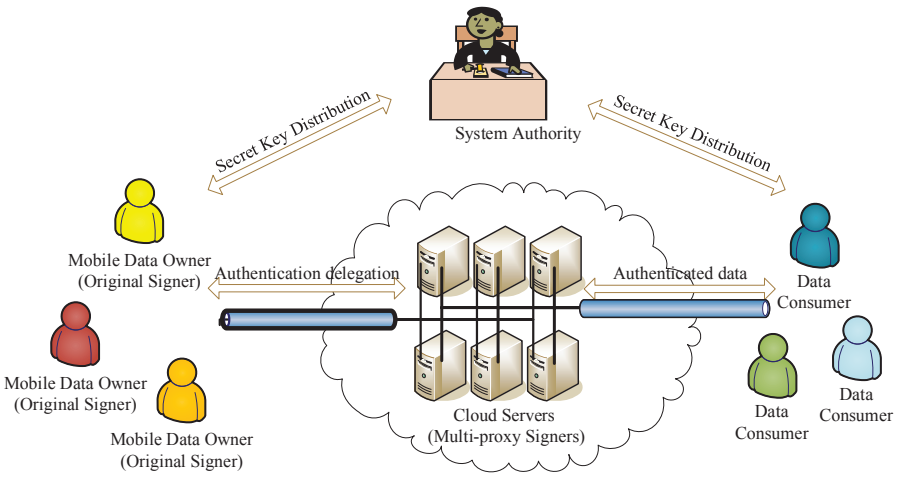

Figure 2. A multi-data owners multi-servers delegated authentication in mobile cloud computing

Since the system initialization and the secret key distribution steps are almost the same as protocols, we only focus on the illustration that the multiple data owners delegate their signing right to multiple cloud servers, which serve the data consumes authentication service.

8.1. Authentication delegation

To delegate the multiple signing capabilities to the various cloud servers, the data owners, as original signers do the following steps to generate a valid warrant ω, which could specify the delegation details.

8.1.1. Delegation generation.

The data owners interactive to confirm the warrant ω by signing it like a multi-signature way.

1) Each data owner $\mathcal{O}_i$ chooses an integer $r_i \in \mathbb{Z}_N^*$ and calculates its cube $R_i = r_i^3 \pmod N$.
2) Each data owner $\mathcal{O}_i$ sets $t_i = H_4(R_i)$ and then broadcasts t_i to other data owners.
3) Once receiving t_i, each $\mathcal{O}_i$ broadcasts R_i to other data owners.
4) Once receiving R_i, each $\mathcal{O}_i$ checks each $t_i \stackrel{?}{=} H_4(R_i)$. If one of the equations dissatisfies, it aborts. Otherwise, $\mathcal{O}_i$ aggregates $R \equiv \prod_{i=1}^n R_i \pmod N$, sets $h_2 = H_2(\omega||R)$, computes $v_i \equiv r_i \cdot d_{\mathcal{O}_i}^{h_2} \pmod N$, and broadcasts v_i to other data owners.
5) Once receiving v_i from other data owners, $\mathcal{O}_i$ checks each Equation (12)

$$v_i^3 \cdot H(ID_{O_i})^{h_2} \stackrel{?}{=} R_i, \tag{12}$$

where $h_2 = H_2(\omega||R)$. If one of the equations dissatisfies, the protocol aborts. After that, $\mathcal{O}_i$ aggregates $V = \prod_{i=1}^n v_i$.

8.1.2. Delegation verification.

Each $\mathcal{P}_i$ confirms $\sigma = (\omega, R, V)$ by checking Equation (13)

$$V^3 \cdot \prod_{i=1}^n H(ID_i)^{h_2} \stackrel{?}{=} R, \tag{13}$$

where $h_2 = H_2(\omega||R)$. Since

$$\begin{aligned} V^3 \cdot \prod_{i=1}^n H(ID_{\mathcal{O}_i})^{h_2} &\equiv \prod_{i=1}^n v_i^3 \cdot H(ID_{\mathcal{O}_i})^{h_2} \equiv \prod_{i=1}^n (r_i \cdot d_i^{h_2})^3 \cdot H(ID_{\mathcal{O}_i})^{h_2} \\ &\equiv \prod_{i=1}^n r_i^3 \cdot \prod_{i=1}^n (d_i^3 \cdot H(ID_{\mathcal{O}_i}))^{h_2} \equiv R \pmod N, \end{aligned} \tag{14}$$

$\mathcal{P}_i$ accepts the delegation only if σ is a valid one.

8.1.3. Proxy key generation.

Once confirming the delegations, each $\mathcal{P}_i$ generates a proxy key by Equation (15)

$$sk_{p_i} = V \cdot d_{\mathcal{P}_i}^{h_3}, \tag{15}$$

where $h_3 = H_3(\omega||R)$. $\mathcal{P}_i$ keeps it on behalf of the data owners when demanded.

8.2. Multi-server authentication generation

Each $\mathcal{P}_i$ signs on the data m under the warrant ω on behalf of a group of the data owners $\mathcal{O}$s as follows.

1) $\mathcal{P}_i$ chooses an integer $\tilde{r}_i \in \mathbb{Z}_N$ and calculates its cube $\tilde{R}_i = \tilde{r}_i^3$ and $\tilde{t}_i = H_4(\tilde{R}_i)$, and then broadcasts $\tilde{t}_i$.
2) Once receiving $\tilde{t}_i$, each $\mathcal{P}_i$ broadcasts $\tilde{R}_i$.
3) Once receiving $\tilde{R}_i$, each $\mathcal{P}_i$ checks each $\tilde{t}_i \stackrel{?}{=} H_4(\tilde{R}_i)$. If one of the equations dissatisfies, the protocol aborts. $\mathcal{P}_i$ aggregates $\tilde{R} \equiv \prod_{i=1}^n \tilde{R}_i \pmod N$, sets $h_5 = H_5(\omega||m||\tilde{R})$, calculates $u_i \equiv \tilde{r}_i \cdot sk_{p_i}^{h_5} \pmod N$, and broadcasts u_i.

DOI: 10.1002/sec
Prepared using secauth.cls

4) Once receiving u_i, $\mathcal{P}_i$ checks each Equation (16)

$$u_i^3 \cdot H(ID_{\mathcal{P}_i})^{h_3 h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_j})^{h_2 h_5} \stackrel{?}{=} \tilde{R}_i \cdot R^{h_5}, \tag{16}$$

where $h_2 = H_2(\omega||R)$, $h_3 = H_3(\omega||R)$ and $h_5 = H_5(\omega||m||\tilde{R})$. If one of the equations dissatisfies, the protocol aborts. $\mathcal{P}_i$ aggregates $U = \prod_{i=1}^{n} u_i$.

When all of the partial signature are correctly checked, the multi-server authentication tag is made up by $p\sigma = (\omega, U, R, \tilde{R})$

8.3. Authentication verification

Once receiving m and $p\sigma = (\omega, U, R, \tilde{R})$, the data consumer makes the following operations.

1) The data consumer checks whether m conforms to ω.
2) The data consumer checks whether $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ are authorized by $\mathcal{O}_1, \mathcal{O}_2, ..., \mathcal{O}_n$ under ω.
3) The data consumer accepts $p\sigma$ if and only if Equation (17) holds:

$$U^3 \cdot \prod_{i=1}^{n} H(ID_{\mathcal{P}_i})^{h_3 h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \stackrel{?}{=} \tilde{R} \cdot R^{h_5} \pmod{N} \tag{17}$$

where $h_2 = H_2(\omega||R)$, $h_3 = H_3(\omega||R)$, and $h_5 = H_5(\omega||m||R)$.

8.4. Correctness

The **MEDAP-II** is correct since

$$\begin{aligned}
& U^3 \cdot \prod_{i=1}^{n} H(ID_{\mathcal{P}_i})^{h_3 h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \equiv \prod_{i=1}^{n} (u_i^3 \cdot H(ID_{\mathcal{P}_i})^{h_3 h_5}) \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \\
\equiv & \prod_{i=1}^{n} ((\tilde{r}_i^3 \cdot sk_{\mathcal{P}_i}^{3h_5}) \cdot H(ID_{\mathcal{P}_i})^{h_3 h_5}) \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \equiv \prod_{i=1}^{n} \tilde{r}_i^3 \cdot (V \cdot d_{\mathcal{P}_i}^{h_3})^{3h_5} \cdot H(ID_{\mathcal{P}_i})^{h_3 h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \\
\equiv & \prod_{i=1}^{n} \tilde{R}_i \cdot V^{3h_5} \cdot (d_{\mathcal{P}_i}^3 \cdot H(ID_{\mathcal{P}_i}))^{h_3 h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \equiv \tilde{R} \cdot V^{3h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \\
\equiv & \tilde{R} \cdot \prod_{j=1}^{n} v_j^{3h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \equiv \tilde{R} \cdot \prod_{j=1}^{n} (r_j \cdot d_{\mathcal{O}_i}^{h_2})^{3h_5} \cdot \prod_{j=1}^{n} H(ID_{\mathcal{O}_i})^{h_2 h_5} \\
\equiv & \tilde{R} \cdot \prod_{j=1}^{n} r_j^{3h_5} \cdot (d_{O_i}^3 \cdot H(ID_{\mathcal{O}_i}))^{h_2 h_5} \equiv \tilde{R} \cdot \prod_{j=1}^{n} R_j^{h_5} \equiv \tilde{R} \cdot R^{h_5} \pmod{N}.
\end{aligned} \tag{18}$$

where we use $d_{ID}^3 \cdot H(ID) \equiv 1 \pmod{N}$ in Section 5.

8.5. Performance Comparison

We compare **MEDAP-I** and **MEDAP-II** with related work, which are secure under different assumptions such as computational Diffie-Hellmen assumptions. We use the operation data from [21] that an operation time for one bilinear pairing operation (denoted as P), map-to-point hash operation (denoted as H), modular-exponentiation (denoted as E), normal scale multiplication (denoted as M), pairing based scalar multiplication (denoted as Psm) in Table I.

From Table II, **MEDAP-I** and **MEDAP-II** are efficient since it does not introduce heavy operations such as the bilinear pairing compared to [22, 18, 23] and are suitable for the cloud environment since the cloud takes the most computation and the data consumers are in a low cost. For simplicity, we denote the symbol $(*)$ in Table II that we assume that the participant number is at least 3.

Table I. Cryptographic operation time (ms) in [19].

Operation	$Pair$	E	H	M	Psm
Running time	20.01	11.20	3.04	0.83	6.38

Table II. Comparison of computation cost and running time when $n = 3$.

Schemes	MPGen	Time	MPSign	Time	MPVerify	Time	Assumption
LC05 [22]	$3P$+$6Psm$	103.71	$3P$+$1E$+$3Psm$	84.57	$4P$+$1E$+$3Psm$	78.19	CDH
CC09 [18]	$3P$+$2H$+$3Psm$	85.34	P+$3H$+$1E$+$2Psm$	127.39	$4P$+$2H$+$3Psm$	99.0	CDH
SP15 [23]	$2P$+$5Psm$	71.98	P+$5Psm$	71.98	$2P$+$4Psm$	65.6	CDH
MEDAP-I	$3E$+$7M$	39.41	$3E$+$(2n$+$5)M$	42.73*	$3E$+$(n$+$4)M$	39.41*	Factorization
MEDAP-II	$3E$+$(3n$+$7)M$	46.88*	$3E$+$(3n$+$5)M$	45.22*	$3E$+$(2n$+$3)M$	41.07*	Factorization

9. RELATED WORK

The cryptographic primitive of proxy signature was introduced by Mambo [24] in which the original signer designates the signing right to an authorized proxy signer and the latter can make a valid signature on behalf of the former. One of the variant of the proxy signature is multi-proxy signature scheme which is proposed by Hwang that an original signer can authorize a *group* of proxy agent to make a signature. Another type is the proxy multi-signature presented by Yi [25] in 2000, which allows that the only one proxy agent can make a signature on behalf of a group of original signers. Currently, most of the proxy signature schemes have been proposed based on bilinear pairing technique [22, 18, 23], however, it brings a heavy overhead to implement bilinear pairing in mobile devices. As a result, the researches have moved to focus on the simpler assumptions and easier implement techniques.

To simplify key management, Cocks [26] presented a provable secure identity-based encryption under quadratic residue assumption without using bilinear pairing in 2001. Bellare [27] proposed the first IBMS scheme proven secure under on RSA assumption with three round interactions. After that, Bagherzandi [28] proposed a two round IBMS also proven secure under RSA assumption by using a commitment scheme. Chai [29] further proposed an identity-based signature scheme based on quadratic residues. Wei [11] proposed two secure IBMS schemes under quadratic residue assumptions using the technique in [27] and [28]. Moreover, Xing [17] proposed an identity-based signature scheme proven secure cubic residues. Wang [15] proposed an improved provably secure identity-based signature based on cubic residues, which is quite efficient compare to [17]. Yu [30] also presented a secure proxy signature scheme from factorization. Wang [16] presented an IBMPS scheme proven secure under cubic residues.

10. CONCLUSION

In this work, an IBMPS scheme has been built under cubic residue assumption, which is as hard as integer factorization problem. We have also proved that our IBMPS scheme is secure against adaptively chosen message attacks and chosen identity/warrant attacks by using random oracle. Aiming to different situations and preventing single point of failure in a mobile cloud computing model, we have proposed three secure multi-entities delegated authentication protocols (MEDAPs), which enable them to conditionally authorize their authentication rights to designated cloud servers without directly exposing their secret keys. After such an authorization, the cloud servers can achieve the data authentication services when the mobile data owners are offline. Furthermore, MEDAPs are efficient compared with other pairing based protocols and the communication overhead is nearly constant which does not depends on the number of cloud servers.

DOI: 10.1002/sec
Prepared using secauth.cls

In our future work, we leave three open problems. Firstly, it needs three rounds of interaction in the algorithm **MPSign**, which costs communication overhead. It is a suggestion to reduce one round by building a commitment scheme under cubic residue assumption like RSA-based commitment [28] and quadratic residues based commitment [11]. Secondly, it might be a solution to propose constructions under higher residues such as 2^k-th residues [31]. Thirdly, the protocol is suggested to reliability and robustness that the cloud system allows authentication service ongoing even if a large proportion of the authorized cloud servers are broken down or compromised.

REFERENCES

1. M. Dong, H. Li, K. Ota, L. T. Yang, and H. Zhu, "Multicloud-based evacuation services for emergency management," *IEEE Cloud Computing*, vol. 1, no. 4, pp. 50–59, 2014.
2. H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.
3. H. Zhu, K.-H. Liu, W. He, and K. Ota, "Quality of experience and quality of protection provisions in emerging mobile networks," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 8–9, 2015.
4. H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," *International Journal of Information Security*, vol. 14, no. 6, pp. 487–497, 2015.
5. L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," *Information Sciences*, vol. 258, pp. 371–386, 2014.
6. J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Transactions on Services Computing*, 2016, dOI: 10.1109/TSC.2016.2520932.
7. C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.
8. J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *INFOCOM 2014*, pp. 2121–2129.
9. J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, B. Waters *et al.*, "Computing on authenticated data," in *TCC 2012*, pp. 1–20.
10. W. Jia, H. Zhu, Z. Cao, L. Wei, and X. Lin, "SDSM: a secure data service mechanism in mobile cloud computing," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS 2011)*, pp. 1060–1065.
11. L. Wei, Z. Cao, and X. Dong, "Secure identity-based multisignature schemes under quadratic residue assumptions," *Security and Communication Networks*, vol. 6, no. 6, pp. 689–701, 2013.
12. Z. Wang, G. Sun, and D. Chen, "A new definition of homomorphic signature for identity management in mobile cloud computing," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 546–553, 2014.
13. J. Yuan and S. Yu, "Flexible and publicly verifiable aggregation query for outsourced databases in cloud," in *IEEE CNS 2013*, pp. 520–524.
14. C. Xiao, W. Jia, H. Zhu, S. Du, and Z. Cao, "Leveraging cloud computing for privacy preserving aggregation in multi-domain wireless networks," in *Wireless Algorithms, Systems, and Applications (WASA 2012)*, pp. 733–744.
15. Z. Wang, L. Wang, S. Zheng, Y. Yang, and Z. Hu, "Provably secure and efficient identity-based signature scheme based on cubic residues," *International Journal of Network Security*, vol. 14, no. 1, pp. 33–38, 2012.
16. F. Wang, C.-C. Chang, C. Lin, and S.-C. Chang, "Secure and efficient identity-based proxy multi-signature using cubic residues," *International Journal of Network Security*, vol. 18, no. 1, pp. 90–98, 2016.
17. D. Xing, Z. Cao, and X. Dong, "Identity based signature scheme based on cubic residues," *Science China Information Sciences*, vol. 54, no. 10, pp. 2001–2012, 2011.
18. F. Cao and Z. Cao, "A secure identity-based multi-proxy signature scheme," *Computers & Electrical Engineering*, vol. 35, no. 1, pp. 86–95, 2009.

19. D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Advances in CryptologyłEUROCRYPT96*, 1996, pp. 387–398.
20. L. Wei, L. Zhang, K. Zhang, and M. Dong, "An efficient and secure delegated multi-authentication protocol for mobile data owners in cloud," in *The 10th Conference on Wireless Algorithms, Systems, and Applications (WASA 2015)*, 2015, pp. 612–622.
21. D. He, J. Chen, and R. Zhang, "An efficient and provably-secure certificateless signature scheme without bilinear pairings," *International Journal of Communication Systems*, vol. 25, no. 11, pp. 1432–1442, 2012.
22. X. Li and K. Chen, "Id-based multi-proxy signature, proxy multi-signature and multi-proxy multi-signature schemes from bilinear pairings," *Applied Mathematics and Computation*, vol. 169, no. 1, pp. 437–450, 2005.
23. R. A. Sahu and S. Padhye, "Provable secure identity-based multi-proxy signature scheme," *International Journal of Communication Systems*, vol. 28, no. 3, pp. 497–512, 2015.
24. M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: delegation of the power to sign messages," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 79, no. 9, pp. 1338–1354, 1996.
25. L. Yi, G. Bai, and G. Xiao, "Proxy multi-signature scheme: a new type of proxy signature scheme," *Electronics Letters*, vol. 36, no. 6, pp. 527–528, 2000.
26. C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues," in *8th IMA International Conference on Cryptography and Coding*, 2001.
27. M. Bellare and G. Neven, "Identity-Based Multi-signatures from RSA," in *the Cryptographers' Track at the RSA Conference (CT-RSA 2007)*, San Francisco, USA, February 5-9, 2007.
28. A. Bagherzandi and S. Jarecki, "Identity-Based Aggregate and Multi-Signature Schemes Based on RSA," in *PKC 2010*, pp. 480–498.
29. Z. Chai, Z. Cao, and X. Dong, "Identity-based signature scheme based on quadratic residues," *Science in China Series F: Information Sciences*, vol. 50, no. 3, pp. 373–380, 2007.
30. Y. Yu, Y. Mu, W. Susilo, Y. Sun, and Y. Ji, "Provably secure proxy signature scheme from factorization," *Mathematical and Computer Modelling*, vol. 55, no. 3, pp. 1160–1168, 2012.
31. M. Joye and B. Libert, "Efficient cryptosystems from 2^k-th power residue symbols." in *EUROCRYPT 2013*, pp. 76–92.

DOI: 10.1002/sec
Prepared using secauth.cls